



Munich Personal RePEc Archive

Ven der Giessen's reordering algorithm in the program for stochastic simulation of econometric models

Carlo Bianchi and Giorgio Calzolari and Remi Doret

IBM Scientific Center, Pisa, Italy

December 1978

Online at <http://mpra.ub.uni-muenchen.de/24880/>

MPRA Paper No. 24880, posted 19. October 2010 08:07 UTC

IBM Italy
PISA SCIENTIFIC CENTER
Z513-5101
December 1978

**VAN DER GIESSEN'S REORDERING ALGORITHM IN THE PROGRAM
FOR STOCHASTIC SIMULATION OF ECONOMETRIC MODELS**

Carlo Bianchi
Giorgio Calzolari
Rémi Doret

IBM Pisa Scientific Center
IBM Pisa Scientific Center
Student at the «Institut National Polytechnique», Grenoble

IBM Scientific Center
Via S. Maria 67
56100 Pisa, Italy

IBM internal use only

This paper describes the application of a reordering algorithm to the equations of econometric models. The algorithm was proposed in 1970 by Van der Giessen [5] and is here applied to the equation format required by the program for stochastic simulation developed at the IBM Scientific Center in Pisa [1], [2].

CONTENTS

| | |
|--|------|
| 1. INTRODUCTION..... | p. 7 |
| 2. INPUT REQUIREMENTS AND EXECUTION PROCEDURE..... | p. 8 |
| 2.1. Input data set..... | p. 8 |
| 2.2. Execution procedure and example of input data set: the Klein-Goldberger model..... | p. 9 |
| 2.3. The output data set..... | p.11 |
| 2.4. Example of output data set: the reordered Klein-Goldberger model..... | p.12 |
| 3. PERFORMANCES..... | p.14 |
| 4. THE PL/1 PROGRAM..... | p.15 |
| 4.1. A simplified flow-chart..... | p.16 |
| 4.2. Program listing..... | p.18 |
| REFERENCES..... | p.26 |

=====

1. INTRODUCTION

The program for stochastic simulation of econometric models proposed in [1] uses the Gauss-Seidel solution algorithm, but does not perform any reordering of the model's equations [2,p.8].

In this paper a program to reorder the equations is proposed. It must be used as a pre-processor; its input data set is the card deck (or disk file) containing the equations of the model, as described in [2,pp.8-9]; the output is a new card deck (or disk file) containing the same equations in a new order. The output data set must then be compiled and successively used as in [2].

The input requirements and the execution procedure are described in some details in section (2.), together with an example on the Klein-Goldberger model. A short comment on the performances of the algorithm is given in section (3.).

Details on the algorithm can be found in the paper by Van der Giessen [5]. In any case, several comment lines are inserted in the PL/1 program, whose listing is printed in section (4.2.), after a simplified flow-chart.

2. INPUT REQUIREMENTS AND EXECUTION PROCEDURE

2.1. Input data set

It is a card image file.

Cards must be written in a standard FORTRAN code; in particular:

- FORTRAN statements must be written one per card within columns 7 through 72.
- A statement may be continued on successive cards (up to 19) by placing any character other than blank in column 6 of each continuation card.
- On the first card of a statement column 6 must be blank.
- For comments, the character C is placed in column 1.

** Statements allowed

- Statements of the form:

$$Y(p) = F(Y(1), \dots, Y(n), \dots)$$

From now on these statements will be called equations. The subscript of an equation is the first number in parentheses (here p).

The subscript of the variable $Y(n)$ is n.

- IF statements

IF (test) followed by an equation

For example:

$$\text{IF}(Y(7) .GT. Y(10)) Y(7) = Y(10)$$

The subscript of the IF statement is the subscript of the equation.

** Other requirements

- Variables to be ordered are always of the form $Y(\dots)$.
Their subscripts form a sequence $1, 2, 3, \dots, n$ with n up to 500.
- Other character strings including 'Y' in their expressions may be used in the

data.

- Blanks may be present everywhere in a card (except in column 7 for statements).
- There are no restrictions about the number of comment cards.
- Several equations and IF statements may have the same subscript.

2.2. Execution procedure and example of input data set:

the Klein-Goldberger model

In this section the FORTRAN code for the Klein-Goldberger revised model [4] is displayed according to the format in [2, pp.8-9]. This card-image data set is the 12-th file on the tape whose contents are described in [2].

| | |
|---|----------|
| SUBROUTINE MODEL (Y,X,NEXO,IREAD,IC,YL,NEND,UMC,A) | KLE00010 |
| IMPLICIT REAL*8 (A-H,O-Z) | KLE00020 |
| DIMENSION Y(1),X(NEXO,IREAD),YL(NEND,IREAD),UMC(1),A(1) | KLE00030 |
| C | KLE00040 |
| C MODEL KLEIN-GOLDBERGER ECONOMETRICA , 1969, APRIL | KLE00050 |
| C | KLE00060 |
| C MEANING OF ENDOGENOUS VARIABLES Y(I) | KLE00070 |
| C | KLE00080 |
| C Y(1) = CD | KLE00090 |
| C Y(2) = CN | KLE00100 |
| C Y(3) = RR | KLE00110 |
| C Y(4) = HH | KLE00120 |
| C Y(5) = IM | KLE00130 |
| C Y(6) = X | KLE00140 |
| C Y(7) = H | KLE00150 |
| C Y(8) = WW | KLE00160 |
| C Y(9) = W | KLE00170 |
| C Y(10) = R | KLE00180 |
| C Y(11) = I | KLE00190 |
| C Y(12) = D | KLE00200 |
| C Y(13) = RS | KLE00210 |
| C Y(14) = PC | KLE00220 |
| C Y(15) = NW | KLE00230 |
| C Y(16) = Y | KLE00240 |
| C Y(17) = P | KLE00250 |
| C Y(18) = SC | KLE00260 |
| C Y(19) = PGR | KLE00270 |
| C Y(20) = PGRR | KLE00280 |
| C | KLE00290 |
| C MEANING OF EXOGENOUS VARIABLES X(I,IC) | KLE00300 |
| C | KLE00310 |
| C X(1,IC) = WG | KLE00320 |
| C X(2,IC) = PM | KLE00330 |
| C X(3,IC) = NG | KLE00340 |
| C X(4,IC) = NS | KLE00350 |
| C X(5,IC) = NL | KLE00360 |
| C X(6,IC) = TC | KLE00370 |
| C X(7,IC) = DU | KLE00380 |
| C X(8,IC) = RD | KLE00390 |
| C X(9,IC) = RE-1 (ONLY LAGGED IN THE MODEL) | KLE00400 |
| C X(10,IC) = G+E | KLE00410 |
| C X(11,IC) = TI | KLE00420 |

```

C X(12,IC) = T KLE00430
C X(13,IC) = SUM KLE00440
C X(14,IC) = CONSTANT KLE00450
C KLE00460
C INVESTMENT FUNCTION (NONRESIDENTIAL) 3.14 KLE00470
  Y(11)=.95*YL(11,IC-1)+A(44)*(YL(6,IC-1)-X(1,IC-1))+ KLE00480
  -A(45)*YL(10,IC-1)+A(46)*YL(11,IC-1)+A(47)*X(14,IC)+UMC(14) KLE00490
C DEPRECIATION EQUATION 3.15 KLE00500
  Y(12)=A(48)*X(13,IC)+A(49)*X(7,IC)+A(50)*X(14,IC)+UMC(15) KLE00510
C INTEREST RATE DETERMINATION EQUATION 3.16 KLE00520
  Y(13)=A(51)*X(8,IC)+A(52)*X(9,IC)+A(53)*X(7,IC)+ KLE00530
  -A(54)*X(14,IC)+UMC(16) KLE00540
C INTEREST RATE STRUCTURE EQUATION 3.10 KLE00550
  Y(10)=A(31)*Y(13)+A(32)*YL(10,IC-1)+A(33)*X(14,IC)+UMC(10) KLE00560
C NATIONAL INCOME NATIONAL PRODUCT IDENTITY 3.18. EXPLIC IN Y KLE00570
  Y(16)=(Y(17)*Y(6)-Y(12)-X(11,IC)-Y(17)*Y(18)- KLE00580
  -X(6,IC)-X(12,IC))/Y(17) KLE00590
C CONSUMPTION FUNCTION (DURABLES) 3.1 KLE00600
  Y(1)=A(1)*(Y(16)-.7*YL(16,IC-1))+(.7+A(2))*YL(1,IC-1)+A(3) KLE00610
  -X(14,IC)+UMC(1) KLE00620
C CONSUMPTION FUNCTION (NONDURABLES) 3.2 KLE00630
  Y(2)=A(4)*Y(16)+A(5)*YL(2,IC-1)+A(6)*X(14,IC)+UMC(2) KLE00640
C INVESTMENT FUNCTION (RESIDENTIAL) 3.3 KLE00650
  Y(3)=A(7)*Y(16)+A(8)*YL(10,IC-1)+A(9)*YL(3,IC-1)+A(10)*X(14,IC) KLE00660
  +UMC(3) KLE00670
C INVESTMENT FUNCTION (INVENTORIES) SCALED BY 10. NORMALIZED FOR H 3.4 KLE00680
  Y(4)=(A(11)/10.*(Y(6)-10.*(-YL(4,IC-1)))+A(12)*YL(4,IC-1) KLE00690
  +A(13)/10*X(14,IC)+UMC(4)/10.)/(1.+A(11)) KLE00700
C IMPORT DEMAND FUNCTION 3.5 KLE00710
  Y(5)=A(14)*Y(6)+A(15)*(X(2,IC)-Y(17))+A(16)*YL(5,IC-1)+ KLE00720
  -A(17)*X(14,IC)+UMC(5) KLE00730
C DEFINITION OF REAL GNP 3.17 KLE00740
  Y(6)=(Y(1)+Y(2)+Y(11)+Y(3) KLE00750
  +10.*(Y(4)-YL(4,IC-1)) KLE00760
  +X(10,IC)-Y(5)) KLE00770
C PRODUCTION FUNCTION WITH SCALING BY 100 IN H. EXPLIC. FOR NW 3.6 KLE00780
C (THE FACTOR 0.230 HAS BEEN EMPIRICALLY CHOSEN FOR CONVERGENCE). KLE00790
  Y(15)=Y(15)+0.230* KLE00800
  -(Y(6)-(X(1,IC)+.95*(YL(6,IC-1)-X(1,IC-1))+A(18)*(Y(11)+Y(3)) KLE00810
  +A(19)*((Y(15)-X(3,IC)+X(4,IC))- .95*(YL(15,IC-1)-X(3,IC-1) KLE00820
  +X(4,IC-1)))+A(20)/100.*(Y(7)-.95*YL(7,IC-1))+A(21)*X(14,IC)+ KLE00830
  -UMC(6))) KLE00840
C HOURS WORKED FUNCTION SCALED BY 100. 3.7 KLE00850
  Y(7)=A(22)*100.*(Y(9)-YL(9,IC-1))+A(23)*100.*(X(5,IC)-Y(15)-X(4,IC) KLE00860
  -)+A(24)*100.*X(14,IC)+UMC(7)*100. KLE00870
C LABOR DEMAND FUNCTION (WAGE SHARE) 3.8 KLE00880
  Y(8)=X(1,IC)+A(25)*(Y(6)-X(1,IC))+A(26)*(YL(8,IC-1)-X(1,IC-1)) KLE00890
  +A(27)*X(14,IC)+UMC(8) KLE00900
C WAGE RATE DETERMINATION EQUATION 3.9 KLE00910
  Y(9)=YL(9,IC-1)+A(28)*(X(5,IC)-Y(15)-X(4,IC))+ KLE00920
  -A(29)*(YL(17,IC-1)-YL(17,IC-2))+A(30)*X(14,IC)+UMC(9) KLE00930
C WAGE IDENTITY NORMALIZED FOR P 3.20 KLE00940
  Y(17)=Y(7)*Y(9)*Y(15)/Y(8)/100. KLE00950
C CORPORATE SAVINGS FUNCTION SCALED BY 10. 3.11. EXPLIC. IN SC KLE00960
  Y(18)=(A(34)*(Y(17)*Y(14)-X(6,IC))+A(35)*(YL(17,IC-1)*YL(14,IC-1) KLE00970
  -X(6,IC-1)-YL(17,IC-1)*YL(18,IC-1))+A(36)*X(14,IC)+UMC(11))/ KLE00980
  -Y(17) KLE00990
C RENTIER INCOME EQUATION 3.13. EXPLIC IN PGRR KLE01000
  Y(20)=(A(40)*Y(17)*(Y(11)+Y(3))+A(41)*(Y(10)-YL(10,IC-1)) KLE01010
  +A(42)*YL(20,IC-1)*YL(17,IC-1)+A(43)*X(14,IC)+UMC(13))/Y(17) KLE01020
C DEFINITION OF PROFIT 3.19. EXPLIC. IN PGR KLE01030
  Y(19)=(Y(17)*Y(6)-Y(12)-X(11,IC)-Y(17)*Y(8)-Y(20)*Y(17))/Y(17) KLE01040
C NONCORPORATE INCOME EQUATION 3.12. EXPLIC. IN PC KLE01050
  Y(14)=Y(19)-10./Y(17)* KLE01060
  -(A(37)/10.*Y(17)*Y(6)+A(38)*(YL(19,IC-1)-YL(14,IC-1))*YL(17,IC-1)/ KLE01070
  -10. KLE01080
  +A(39)/10.*X(14,IC)+UMC(12)/10.) KLE01090
C ***** KLE01100
  RETURN KLE01110
  END KLE01120

```

To process the reordering algorithm, first of all the first 46 cards of the data set must be dropped, stored into a separate file, as well as the RETURN and END statements. The comment card

C*****

is mandatory at the end of the data set and must appear nowhere else.

After these preliminary operations, INPUT FORTRAN A1 must be the CMS identifier of the data set [3]; the following VANDERG EXEC procedure

```
FILEDEF ONE DISK INPUT FORTRAN A1 (RECFM F LRECL 80 BLOCK 80)
FILEDEF TWO DISK OUTPUT FORTRAN A1 (RECFM F LRECL 80 BLOCK 80)
GLOBAL TXTLIB PLILIB
LOAD VANDERG
START
```

can now be used issuing the CMS command:

vanderg

2.3. The output data set

The output processing is performed by the subroutines ECRIT and IMPRIM (see section 4.). The equations are written following the sequence (put in SEQ) found by Van der Giessen's method.

All the comments preceding a statement in the input file are present in the output file immediately before the statement itself.

Several statements may have the same subscript. In the output file they are written in a block (inserted at the place indicated by SEQ). Inside the block they have the same sequence as their relative order in the input file.

The comment 'END OF RECURSIVE PART' is written after the end of the recursive equations (determined in subroutine RNG), if any.

The comment 'END OF SIMULTANEOUS PART' is put just before the beginning of the post-recursive part (if any).

For further details see the program listing (MAIN and subroutine RNG, section 4.2.).

2.4. Example of output data set:

the reordered Klein-Goldberger model

The result of the execution of the VANDERG EXEC procedure is the creation of the file whose CMS identifier is OUTPUT FORTRAN, containing the reordered equations and the corresponding comment lines as follows:

```

C INVESTMENT FUNCTION (NONRESIDENTIAL) 3.14 OUT00010
  Y(11)=.95*YL(11,IC-1)+A(44)*(YL(6,IC-1)-X(1,IC-1))+
  +A(45)*YL(10,IC-1)+A(46)*YL(11,IC-1)+A(47)*X(14,IC)+UMC(14) OUT00020
C DEPRECIATION EQUATION 3.15 OUT00030
  Y(12)=A(48)*X(13,IC)+A(49)*X(7,IC)+A(50)*X(14,IC)+UMC(15) OUT00040
C INTEREST RATE DETERMINATION EQUATION 3.16 OUT00050
  Y(13)=A(51)*X(8,IC)+A(52)*X(9,IC)+A(53)*X(7,IC)+
  +A(54)*X(14,IC)+UMC(16) OUT00060
C INTEREST RATE STRUCTURE EQUATION 3.10 OUT00070
  Y(10)=A(31)*Y(13)+A(32)*YL(10,IC-1)+A(33)*X(14,IC)+UMC(10) OUT00080
C**** END OF RECURSIVE PART ***** OUT00090
C WAGE RATE DETERMINATION EQUATION 3.9 OUT00100
  Y(9)=YL(9,IC-1)+A(28)*(X(5,IC)-Y(15)-X(4,IC))+
  +A(29)*(YL(17,IC-1)-YL(17,IC-2))+A(30)*X(14,IC)+UMC(9) OUT00110
C HOURS WORKED FUNCTION SCALED BY 100. 3.7 OUT00120
  Y(7)=A(22)*100.*(Y(9)-YL(9,IC-1))+A(23)*100.*(X(5,IC)-Y(15)-X(4,IC)+
  +A(24)*100.*X(14,IC)+UMC(7)*100. OUT00130
C INVESTMENT FUNCTION (INVENTORIES) SCALED BY 10. NORMALIZED FOR H 3.4 OUT00140
  Y(4)=(A(11)/10.*(Y(6)-10.*(-YL(4,IC-1)))+A(12)*YL(4,IC-1)
  ++A(13)/10*X(14,IC)+UMC(4)/10.)/(1.+A(11)) OUT00150
C LABOR DEMAND FUNCTION (WAGE SHARE) 3.8 OUT00160
  Y(8)=X(1,IC)+A(25)*(Y(6)-X(1,IC))+A(26)*(YL(8,IC-1)-X(1,IC-1))
  ++A(27)*X(14,IC)+UMC(8) OUT00170
C WAGE IDENTITY NORMALIZED FOR P 3.20 OUT00180
  Y(17)=Y(7)*Y(9)*Y(15)/Y(8)/100. OUT00190
C IMPORT DEMAND FUNCTION 3.5 OUT00200
  Y(5)=A(14)*Y(6)+A(15)*(X(2,IC)-Y(17))+A(16)*YL(5,IC-1)+
  +A(17)*X(14,IC)+UMC(5) OUT00210
C DEFINITION OF PROFIT 3.19. EXPLIC. IN PGR OUT00220
  Y(19)=(Y(17)*Y(6)-Y(12)-X(11,IC)-Y(17)*Y(8)-Y(20)*Y(17))/Y(17) OUT00230
C NONCORPORATE INCOME EQUATION 3.12. EXPLIC. IN PC OUT00240
  Y(14)=Y(19)-10./Y(17)*
  +A(37)/10.*Y(17)*Y(6)+A(38)*(YL(19,IC-1)-YL(14,IC-1))*YL(17,IC-1)/
  +10. OUT00250
  ++A(39)/10.*X(14,IC)+UMC(12)/10.) OUT00260
C CORPORATE SAVINGS FUNCTION SCALED BY 10. 3.11. EXPLIC. IN SC OUT00270
  Y(18)=(A(34)*(Y(17)*Y(14)-X(6,IC))+A(35)*(YL(17,IC-1)*YL(14,IC-1)
  +-X(6,IC-1)-YL(17,IC-1)*YL(18,IC-1))+A(36)*X(14,IC)+UMC(11))/
  +Y(17) OUT00280
C NATIONAL INCOME NATIONAL PRODUCT IDENTITY 3.18. EXPLIC IN Y OUT00290
  Y(16)=(Y(17)*Y(6)-Y(12)-X(11,IC)-Y(17)*Y(8)-
  +X(6,IC)-X(12,IC))/Y(17) OUT00300
C CONSUMPTION FUNCTION (DURABLES) 3.1 OUT00310
  Y(1)=A(1)*(Y(16)-.7*YL(16,IC-1))+(.7+A(2))*YL(1,IC-1)+A(3)
  +X(14,IC)+UMC(1) OUT00320
C CONSUMPTION FUNCTION (NONDURABLES) 3.2 OUT00330
  Y(2)=A(4)*Y(16)+A(5)*YL(2,IC-1)+A(6)*X(14,IC)+UMC(2) OUT00340
C INVESTMENT FUNCTION (RESIDENTIAL) 3.3 OUT00350
  Y(3)=A(7)*Y(16)+A(8)*YL(10,IC-1)+A(9)*YL(3,IC-1)+A(10)*X(14,IC)
  ++UMC(3) OUT00360
C DEFINITION OF REAL GNP 3.17 OUT00370
  Y(6)=(Y(1)+Y(2)+Y(11)+Y(3)
  ++10.*(Y(4)-YL(4,IC-1))
  ++X(10,IC)-Y(5)) OUT00380
C PRODUCTION FUNCTION WITH SCALING BY 100 IN H. EXPLIC. FOR NW 3.6 OUT00390
C (THE FACTOR 0.230 HAS BEEN EMPIRICALLY CHOSEN FOR CONVERGENCE). OUT00400

```

```

      Y(15)=Y(15)+0.230*
      +(Y(6)-(X(1,IC)+.95*(YL(6,IC-1)-X(1,IC-1))+A(18)*(Y(11)+Y(3))
      ++A(19)*((Y(15)-X(3,IC)+X(4,IC))-.95*(YL(15,IC-1)-X(3,IC-1)
      ++X(4,IC-1)))+A(20)/100.*(Y(7)-.95*YL(7,IC-1))+A(21)*X(14,IC)+
      +UMC(6)))
C  RENTIER INCOME EQUATION 3.13. EXPLIC IN PGRR
      Y(20)=(A(40)*Y(17)*(Y(11)+Y(3))+A(41)*(Y(10)-YL(10,IC-1))
      ++A(42)*YL(20,IC-1)*YL(17,IC-1)+A(43)*X(14,IC)+UMC(13))/Y(17)

```

```

      OUT00570
      OUT00580
      OUT00590
      OUT00600
      OUT00610
      OUT00620
      OUT00630
      OUT00640

```

The heading statements previously dropped, as well as the RETURN and END statements, must be reinserted.

The data set so obtained (OUTPUT FORTRAN) must be renamed KLEINGOL FORTRAN and compiled; simulation can then be performed as in {2,p.29}.

3. PERFORMANCES

The solution of the non-reordered Klein-Goldberger model for one year (for example 1965), with a relative tolerance $0.1E-06$ and assuming the historical values of the endogenous variables as starting values of the Gauss-Seidel procedure, requires 42 iterations.

The solution of the model, reordered as in section (2.4.), requires 18 iterations.

The solution of the Klein-I model, as in [2, pp.8-9], at 1941 requires 25 iterations. After reordering, the same model requires 18 iterations.

The reordering of the Klein-Goldberger model requires about 8 seconds of CPU time on a computer IBM/370 model 168.

=====

4. THE PL/1 PROGRAM

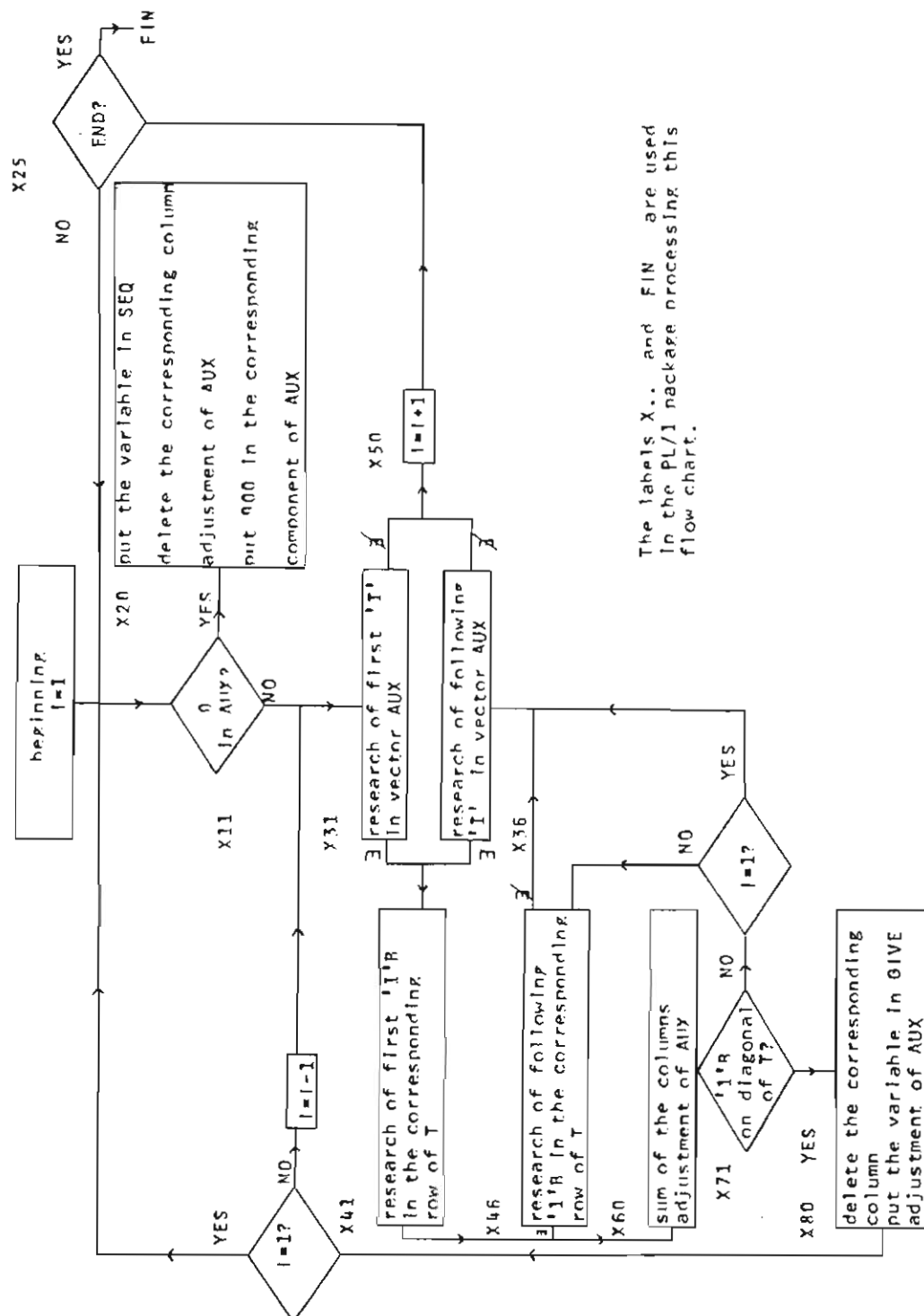
Whenever possible, within the program listing and in the flow-charts the same variables names as in Van der Giessen's paper [5] are maintained.

The program:

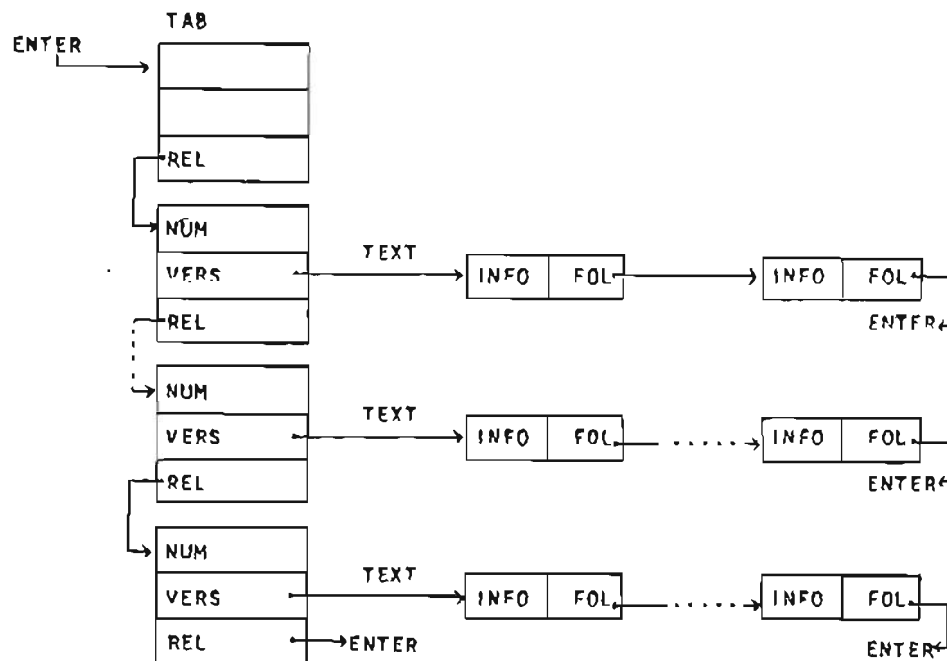
- reads the data set,
- fills the structure matrix T after a syntactic analysis (research of the sequence Y(...)) of each equation,
- stores the cards in a lists structure (see later in this paragraph),
- processes the Van der Giessen's algorithm,
- fills the output file with the new sequence.

Details about each subroutine are given in the listing of the program (see 4.2.).

4.1. A simplified flow-chart



In order to have a simple output processing, the cards are stored in this way (a list of lists is used):



The first element of TAB is just created to begin the main list.

In each list TEXT we have a statement (equation or IF statement) with the comment cards coming just before this statement. Each INFO contains one card of the statement. NUM is the subscript of the equation or IF statement.

If several statements have the same subscript, the corresponding row in T depends on all the variables found in all the statements.


```

      /* AFTER CANCELLING THE BLANKS 'K' IS ON THE LAST NONBLANK */
      /* CHARACTER OF A CARD */
      DO WHILE(SUBSTR(CARD,K,1)=' ');K=K-1;END;
      /* WE PUT THE FIRST CARD OF THE EQUATION IN 'EQU' */
      EQU=SUBSTR(CARD,7,K-6);
      LENG(2)=K-6;J=K-6;
      READ FILE(ONE) INTO (CARD);
      I=3;
      /* THE FOLLOWING LOOP FINDS THE REST OF CARDS COMPOSING AN */
      /* EQUATION AND PUT THEM IN 'EQU' */
      DO WHILE((SUBSTR(CARD,1,1)~='C')&(SUBSTR(CARD,6,1)~=' '));
      K=72;
      DO WHILE(SUBSTR(CARD,K,1)=' ');K=K-1;END;
      EQU=SUBSTR(EQU,1,J)||SUBSTR(CARD,7,K-6);
      LENG(1)=K-6;
      I=I+1;J=J+K-6;
      READ FILE(ONE) INTO (CARD);
      END;
END CREATE;

/*****
/* WHEN WE HAVE FOUND A STRING 'Y(...)', 'COD' LOOKS FOR */
/* THE INDEX BETWEEN PARENTHESES AND PUT IT IN 'Z1' */
*****/
COD: PROC(L);
/*-----*/
DCL Z4 CHAR(2);
DCL (L,K) PIC'9999';
K=L;
      /* WE LOOK FOR RIGHT PARENTHESIS OF Y( ) */
      DO WHILE(SUBSTR(EQU,K,1)~=')');K=K+1;END;
      K=K-1;
      /* WE ARE BETWEEN THE PARENTHESES OF Y( ) AND WE LOOK FOR */
      /* THE FIRST DIGIT OF THE INDEX STARTING FROM THE RIGHT */
      DO WHILE(SUBSTR(EQU,K,1)=' ');K=K-1;END;
      /* WE EXAMINE IF THERE ARE ONE OR TWO DIGITS IN INDEX */
      /* AND FILL 'Z1' CONSEQUENTLY */
      IF((SUBSTR(EQU,K-1,1)=' ')|(SUBSTR(EQU,K-1,1)='('))
      THEN Z1='00'||SUBSTR(EQU,K,1);
      ELSE IF((SUBSTR(EQU,K-2,1)=' ')|(SUBSTR(EQU,K-2,1)='('))
      THEN Z1='0'||SUBSTR(EQU,K-1,2);
      ELSE Z1=SUBSTR(EQU,K-2,3);
      END COD;

/*****
/* THIS SUBROUTINE FILLS A ROW OF STRUCTURE ARRAY 'T' */
/* THE EQUATION IS DEAL WITH IS IN 'EQU' */
*****/
STUD: PROC(M3);
/*-----*/
DCL(M1,M2,M3)PIC'9999',Z2 CHAR(1),
COL PIC'999';
      /* COD(M3) GIVES US THE ROW TO FILL IN MATRIX 'T' */
      CALL COD(M3);ROW=Z1;LENG(1)=ROW;M1=M3+3;
      DO WHILE(M1<J); /* J INDICATES THE END OF 'EQU' */
      IF(SUBSTR(EQU,M1,1)='Y') THEN
      /* WE HAVE FOUND THE CHARACTER 'Y', WE LOOK IF THERE IS */
      /* THE FOLLOWING SEQUENCE : Y(....) */
      DO;M2=M1+1;
      DO WHILE (SUBSTR(EQU,M2,1)=' ');M2=M2+1;END;
      IF( SUBSTR(EQU,M2,1)='(') THEN
      DO ;
      Z2=SUBSTR(EQU,M1-1,1);
      IF((Z2=' ')|(Z2='*')|(Z2='+')|(Z2='-')|(Z2='(')
      |(Z2='/')|(Z2='=')) THEN
      /* WE HAVE FOUND A STRING 'Y(...)' IN EQU. 'COD' GIVES THE */
      /* COLUMN INDEX 'COL' AND WE DO T(ROW,COL)='1'B */
      DO ;

```

```

CALL COD(M1);
COL=21;T(ROW,COL)='1'B;
M1=M1+3;
END;
END;
MI=M1+1;
END;
END STUD ;

/*****
/* 'ECRIT' PRINTS THE EQUATION WHOSE INDEX IS ORD(Y(ORD)=....) */
/* IN THE SAME WAY AS IT WAS GIVEN IN DATA */
*****/
/*****
/*-----*/
      ECRIT:PROC(ORD,RANG);
/*-----*/
DCL RANG POINTER ; /* USED TO PRINT SEVERAL LIST HAVING SAME INDEX */
DCL ORD PIC'999';
COUR1=RANG; /* RESEARCH OF INDEX ORD BEGINS AT ADDRESS RANG */
DO WHILE((COUR1->NUM=ORD)&(COUR1~ENTER));
COUR1=COUR1->REL;
END;
IF(COUR1=ENTER) THEN GOTO TERM;
COUR2=COUR1->VERS;
/* WE ARE ON THE RIGHT ADDRESS, WE WRITE THE FIRST CARD */
WRITE FILE(TWO) FROM(COUR2->INFO);
/* THE FOLLOWING LOOP WRITES THE REST OF TEXT */
DO WHILE(COUR2->FOL~ENTER);
COUR2=COUR2->FOL;
WRITE FILE(TWO) FROM(COUR2->INFO);
END;
TERM:END ECRIT;

/*****
/* THIS SUBROUTINE STORES AN EQUATION IN THE SAME WAY AS
/* IT WAS READ IN ORDER TO HAVE A SIMPLE PRINTING. THIS IS
/* DONE WITH A LIST STRUCTURE
*****/
/*****
/*-----*/
      ST:PROC;
/*-----*/
DCL CARD1 CHAR(80),
POS PIC'9999', /* INDICATES POSITION IN STRING 'EQU' */
R PIC'99';
/* PROCESSING OF THE FIRST CARD OF AN EQUATION OR IF STATEMENT.
/* FIRSTLY WE EXAMINE IF COMMENTS HAVE BEEN ALREADY PUT
/* IN A LIST TEXT : IF COM='0'B (NO COMMENTS) WE CREATE A NEW
/* LIST TEXT(Y1) OTHERWISE WE CONTINUE THE LIST ALREADY BEGUN(Y2)
CARD1=' ' ;
CARD1=' ' || SUBSTR(EQU,1,LENG(2));
IF (COM='0'B ) THEN GOTO Y1;
ELSE GOTO Y2;
Y2: COUR1->NUM=LENG(1);
ALLOCATE TEXT SET (P);
COUR2->FOL=P;
P->INFO=CARD1;
P->FOL=ENTER;
COUR2=P;
POS=LENG(2)+1;
GOTO Y3;
Y1: ALLOCATE TAB SET (Q);
COUR1->REL=Q;
Q->REL=ENTER;
Q->NUM=LENG(1);
COUR1=Q;
ALLOCATE TEXT SET(P);
COUR2=P;
Q->VERS=P;
P->INFO=CARD1;

```

VANO1370
VANO1380
VANO1390
VANO1400
VANO1410
VANO1420
VANO1430
VANO1440
VANO1450
VANO1460
VANO1470
VANO1480
VANO1490
VANO1500
VANO1510
VANO1520
VANO1530
VANO1540
VANO1550
VANO1560
VANO1570
VANO1580
VANO1590
VANO1600
VANO1610
VANO1620
VANO1630
VANO1640
VANO1650
VANO1660
VANO1670
VANO1680
VANO1690
VANO1700
VANO1710
VANO1720
VANO1730
VANO1740
VANO1750
VANO1760
VANO1770
VANO1780
VANO1790
VANO1800
VANO1810
VANO1820
VANO1830
VANO1840
VANO1850
VANO1860
VANO1870
VANO1880
VANO1890
VANO1900
VANO1910
VANO1920
VANO1930
VANO1940
VANO1950
VANO1960
VANO1970
VANO1980
VANO1990
VANO2000
VANO2010
VANO2020
VANO2030
VANO2040
VANO2050
VANO2060

```

P->FOL=ENTER;
POS=LENG(2)+1;
Y3:R=3;
/* LOOP TO CONTINUE THE LIST 'TEXT' UNTIL THE END OF
/* THE EQUATION
DO WHILE (POS<J);
CARD1=' ';
CARD1=' '||SUBSTR(EQU,POS,LENG(R));
ALLOCATE TEXT SET(P);
COUR2->FOL=P;
P->INFO=CARD1;
P->FOL=ENTER;
COUR2=P;
POS=POS + LENG(R);
R=R+1;
END;
END ST ;

/*****
/* THIS PROCEDURE RECOGNIZES IF WE ARE ON A STATEMENT
/* Y( )= OR AN 'IF STAT'( IF(.....) ).AFTER THERE
/* IS A SPECIAL TREATMENT FOR EACH KIND OF STATEMENT
*****/
/-----*/
TESTY:PROC ;
/-----*/
DCL PEQU PIC '9999',/* POINTER IN STRING EQU
NPC PIC '999'; /* USED TO TEST A NUMBER OF PARENTHESIS
IF (SUBSTR(EQU,1,1)='Y') THEN
/* WE ARE ON AN 'IF STAT'
DO;
PEQU=1;
/* WE LOOK FOR THE LEFT PARENTHESIS OF Y( )
DO WHILE (SUBSTR(EQU,PEQU,1)='('); PEQU=PEQU+1; END;
/* PEQU IS ON THE FIRST PARENTHESIS
NPC=1 ; PEQU = PEQU + 1;
DO WHILE (NPC<=0) ;
/* WE LOOK FOR THE END OF TEST IF BY COUNTING THE PARENTHESSES
/* WITH NPC
IF (SUBSTR(EQU,PEQU,1)='(') THEN NPC = NPC +1;
ELSE IF (SUBSTR(EQU,PEQU,1)=')') THEN NPC = NPC -1;
PEQU=PEQU+1;
END;
PEQU = PEQU +1;
/* WE LOOK FOR THE FIRST CHAR '-' FOLLOWING THE END OF TEST
DO WHILE (SUBSTR(EQU,PEQU,1)=' '); PEQU = PEQU +1; END;
IF ((SUBSTR(EQU,PEQU,2)='Y(')|(SUBSTR(EQU,PEQU,2)='Y ')) THEN
/* THE STAT FOLLOWING IF(..) IS Y( )=.. WE HAVE TO TREAT
/* EQUATION IN STUD(PEQU) PEQU IS NOW ON 'Y' OF Y( )
CALL STUD(PEQU);
ELSE
/* THERE IS NOT Y( )= AFTER 'IF (..)' SO WE HAVE JUST TO
/* STORE THIS STATE WITH 999 IN 'NUM' OF LIST 'TAB'
/* WHICH MEANS THAT THIS STATE WILL BE PRINTED AT THE END
LENG(1)=999;
END;
ELSE DO;
/* THE STATEMENT IS 'Y(..)=.... ' WE TREAT IT IN STUD (1)
CALL STUD(1) ; N=N+1 ; END ;
END TESTY ;

/*****
/* ORDERING OF THE EQUATIONS FOLLOWING 'VAN DER GIESSEN METHOD'
*****/
/-----*/
RNG:PROC;
/-----*/
DCL IT BIT(1);
DCL A PIC'999', /* POINTER IN VECTOR 'AUX'
G PIC'999', /* POINTER IN VECTOR 'GIVE'

```

VAN02070
VAN02080
VAN02090
VAN02100
VAN02110
VAN02120
VAN02130
VAN02140
VAN02150
VAN02160
VAN02170
VAN02180
VAN02190
VAN02200
VAN02210
VAN02220
VAN02230
VAN02240
VAN02250
VAN02260
VAN02270
VAN02280
VAN02290
VAN02300
VAN02310
VAN02320
VAN02330
VAN02340
VAN02350
VAN02360
VAN02370
VAN02380
VAN02390
VAN02400
VAN02410
VAN02420
VAN02430
VAN02440
VAN02450
VAN02460
VAN02470
VAN02480
VAN02490
VAN02500
VAN02510
VAN02520
VAN02530
VAN02540
VAN02550
VAN02560
VAN02570
VAN02580
VAN02590
VAN02600
VAN02610
VAN02620
VAN02630
VAN02640
VAN02650
VAN02660
VAN02670
VAN02680
VAN02690
VAN02700
VAN02710
VAN02720
VAN02730
VAN02740
VAN02750
VAN02760

```

S PIC'999', /* POINTER IN VECTOR 'SEQ' */
K PIC'999', /* POINTER IN THE CURRENT ROW OF 'T' */
(I,J) PIC'999' ;
/* INITIATION OF VECTOR 'AUX' */
A=1;
DO WHILE(A<=N);
G=0;S=1;
DO WHILE(S<=N);
IF(T(A,S)='1'B) THEN G=G+1;
S=S+1;
END;AUX(A)=G;A=A+1;END;
/* WE WANT TO SEPARATE THE RECURSIVE OF THE SYSTEM FROM THE */
/* SIMULTANEOUS PART. IN MATRIX 'T', 'RECURSIVE VARIABLES' */
/* HAVE NO '1'B IN THEIR COLUMN. IN THE FOLLOWING LOOP WE */
/* LOOK FOR SUCH VARIABLES AND PUT THEM IN VECTOR 'REC'. */
R1=1; REC(1)=1;
DO A=1 TO N;
G=0;S=1;
DO WHILE(S<=N);
IF (T(S,A)='1'B) THEN G=G+1;
S=S+1;
END;
IF G=0 THEN DO;
REC(R1)=A; R1=R1+1;END;
END;
I=1;G=1;S=1;
IT='0'B;
X10=A=1;
/* X11 TEST TO FIND '0' IN VECTOR 'AUX' WHICH MEANS THAT
/* THE CORRESPONDING VARIABLE MUST BE PUT IN 'SEQ'
X11: IF (AUX(A)=0) THEN GOTO X20;
A=A+1;
IF (A<=N) THEN GOTO X11;
/* WE WANT TO SEPARATE VARIABLES INDEPENDANT FROM ENDOGENOUS
/* VARIABLES FROM THE OTHER VARIABLES. WE KNOW THAT THESE
/* VARIABLES ARE PUT IN 'SEQ' AT THE BEGINNING OF THE ALGO
/* RITHM(IT='0'B). SO WE CAN EASILY FIND THIS SEQUENCE OF
/* VARIABLES AND THEN WRITE A COMMENT OF OUR CHOICE IN THE
/* RIGHT PLACE OF THE OUTPUT FILE
IF (IT='0'B) THEN
DO;
IF (S>1) THEN
DO;
ALLOCATE TAB SET(Q);
COUR1->REL=Q; Q->REL=ENTER; Q->NUM=SEQ(S-1); COUR1=Q;
ALLOCATE TEXT SET(P);
COUR2=P;
Q->VERS=P;
P->INFO='C**** END OF RECURSIVE PART ****';
P->FOL=ENTER;
END;
END;
IT='1'B;
/* X30 RESEARCH OF THE FIRST 1 IN VECTOR 'AUX' */
X30:A=1;
X31: IF (AUX(A)=1) THEN GOTO X40;
A=A+1;
IF (A<=N) THEN GOTO X31;
X50:I=I+1;
GOTO X25;
/* X40 RESEARCH OF THE FIRST 1 IN ROW 'A' OF THE MATRIX 'T'
X40:K=1;
X41: IF (T(A,K)='1'B) THEN GOTO X60;
K=K+1;
IF (K<=N) THEN GOTO X41;
/* WE HAVE FOUND A '1'B IN COLUMN K ROW A. X60 WE ADD COL 'A' TO
/* COL 'K' AND ALSO MODIFY 'AUX' AFTER THIS OPERATION
X60: DO B=1 TO N;
IF((T(B,K)='0'B)&(T(B,A)='1'B)) THEN
DO;

```

```

      AUX(B)=AUX(B)+1;
      T(B,K)='1'B;
      END;
      /* RESEARCH OF A '1'B IN THE DIAGONAL OF MATRIX 'T' */
X70: J=1;
X71: IF (T(J,J)='1'B) THEN GOTO X80;
      J=J+1;
      IF (J<=N) THEN GOTO X71;
      IF (I=1) THEN GOTO X35;
      /* WE HAVE ALREADY FOUND ONE OR MORE '1'B IN ROW A. THEIR
      /* TREATMENT HAS BEEN DONE. WE RESEARCH THE FOLLOWING '1'B. */
X45: K=K+1;
X46: IF (T(A,K)='1'B) THEN GOTO X60;
      K=K+1;
      IF (K<=N) THEN GOTO X46;
      /* WE HAVE ALREADY TREATED ONE OR MORE '1' IN VECTOR
      /* AUX. WE RESEARCH THE FOLLOWING '1'
X35: A=A+1;
X36: IF (AUX(A)=1) THEN GOTO X40;
      A=A+1;
      IF (A<=N) THEN GOTO X36;
      GOTO X50;
      /* WE HAVE FOUND A VARIABLE TO BE GIVEN. SO WE DELETE THE
      /* CORRESPONDING COLUMN J, WE PUT THIS VARIABLE IN 'GIVE'
      /* AND ADJUST 'AUX'
X80: A=1;
X81: IF (T(A,J)='0'B) THEN GOTO X82;
      T(A,J)='0'B;
      AUX(A)=AUX(A)-1;
X82: A=A+1;
      IF (A<=N) THEN GOTO X81;
      GIVE(G)=J;
      G=G+1;
      IF (I=1) THEN GOTO X10;
      I=I-1;
      GOTO X30;
      /* WE HAVE FOUND A ROW 'A' WITH AUX(A)=0. WE PUT THIS VARIABLE
      /* IN VECTOR 'SEQ', WE DELETE THE CORRESPONDING COLUMN IN
      /* MATRIX 'T' AND ADJUST 'AUX'
X20: SEQ(S)=A;
      AUX(A)=900;
      S=S+1;
      J=1;
X21: IF (T(J,A)='0'B) THEN GOTO X22;
      T(J,A)='0'B;
      AUX(J)=AUX(J)-1;
X22: J=J+1;
      IF (J<=N) THEN GOTO X21;
      /* X25 TEST TO KNOW IF THE SEQUENCE IS FINISHED */
X25: IF (S=N+1) THEN GOTO FIN;
      ELSE GOTO X10;
FIN: END RNG;

/*****
/* THIS SUBROUTINE SEARCHS IF 'IN' IS ONE OF THE COMPONENTS OF
/* VECTOR 'REC'. IF YES TROUVE=1 ELSE TROUVE=0
/*****
/*****
COMP: PROC(IN);
/*****
DCL IN PIC'999',
      R2 PIC'999',
      TROUVE PIC'9';
TROUVE=0; R2=1;
DO WHILE (R2<R1);
  IF (REC(R2)=IN) THEN DO; TROUVE=1; R2=R1; END;
  ELSE R2=R2+1;
END;
RETURN(TROUVE);

```

VAN03470
 VAN03480
 VAN03490
 VAN03500
 VAN03510
 VAN03520
 VAN03530
 VAN03540
 VAN03550
 VAN03560
 VAN03570
 VAN03580
 VAN03590
 VAN03600
 VAN03610
 VAN03620
 VAN03630
 VAN03640
 VAN03650
 VAN03660
 VAN03670
 VAN03680
 VAN03690
 VAN03700
 VAN03710
 VAN03720
 VAN03730
 VAN03740
 VAN03750
 VAN03760
 VAN03770
 VAN03780
 VAN03790
 VAN03800
 VAN03810
 VAN03820
 VAN03830
 VAN03840
 VAN03850
 VAN03860
 VAN03870
 VAN03880
 VAN03890
 VAN03900
 VAN03910
 VAN03920
 VAN03930
 VAN03940
 VAN03950
 VAN03960
 VAN03970
 VAN03980
 VAN03990
 VAN04000
 VAN04010
 VAN04020
 VAN04030
 VAN04040
 VAN04050
 VAN04060
 VAN04070
 VAN04080
 VAN04090
 VAN04100
 VAN04110
 VAN04120
 VAN04130
 VAN04140
 VAN04150
 VAN04160


```

END COMP;

/*****
/* THIS SUBROUTINE WRITES IN THE OUTPUT FILE ALL THE LISTS 'TEXT' */
/* HAVING THE SAME INDEX 'NUM' IN THE LIST 'TAB'. */
*****/
INPRIM:PROC(INDIC);
/*****
DCL INDIC PIC'999';
RANG1=ENTER->REL;
/* WE LOOK IF THERE IS MORE THAN ONE LIST WITH THE */
/* SAME 'NUM' AND WE CALL ECRIT AS MANY TIMES AS NECESSARY */
DO WHILE(RANG1=ENTER);
CALL ECRIT(INDIC,RANG1);
IF (COUR1=ENTER) THEN RANG1=COUR1->REL;
ELSE RANG1=ENTER;
END;
END INPRIM;

/*****
/*
MAIN PROCEDURE
*****/

/* INITIATION */
ALLOCATE TAB SET(Q);
ENTER=Q;
COUR1=Q;
Q->REL=ENTER;
T='0'B;N=0;
COM='0'B;
OPEN FILE(ONE) INPUT;
OPEN FILE(TWO) OUTPUT;
READ FILE(ONE) INTO (CARD);
/* LOOP TO FIND THE EQUATIONS, STORE THEM AND FILL MATRIX 'T' */
ON ENDFILE(ONE) GOTO SUIT ;
LOOP:IF (SUBSTR(CARD,1,1)='C') THEN
DO;
/* WE ARE ON A COMMENT CARD.COM='0'B MEANS THAT */
/* THIS IS THE FIRST COMMENT CARD COMING AFTER */
/* AN EQUATION OR IF STATEMENT SO WE CREATE */
/* A NEW LIST TEXT AND BEGIN IT WITH THIS CARD */
IF (COM='0'B) THEN
DO;
ALLOCATE TAB SET(Q);
COUR1->REL=Q; Q->REL=ENTER;COUR1=Q; COM='1'B;
ALLOCATE TEXT SET(P);
COUR2=P; Q->VERS=P; P->INFO=CARD; P->FOL=ENTER;
END;
ELSE
/* WE CONTINUE THE 'COMMENT LIST ' ALREADY BEGUN */
DO;
ALLOCATE TEXT SET(P);
COUR2->FOL=P;
P->INFO=CARD;
P->FOL=ENTER;
COUR2=P;
END;
READ FILE(ONE) INTO (CARD) ;
END;
ELSE
/* WE ARE NOT ON A COMMENT SO WE HAVE TO DO THE WHOLE TREATMENT */
DO;
CALL CREATE;
CALL TESTY;
CALL ST;
COM='0'B;
END;

```

VAN04170
VAN04180
VAN04190
VAN04200
VAN04210
VAN04220
VAN04230
VAN04240
VAN04250
VAN04260
VAN04270
VAN04280
VAN04290
VAN04300
VAN04310
VAN04320
VAN04330
VAN04340
VAN04350
VAN04360
VAN04370
VAN04380
VAN04390
VAN04400
VAN04410
VAN04420
VAN04430
VAN04440
VAN04450
VAN04460
VAN04470
VAN04480
VAN04490
VAN04500
VAN04510
VAN04520
VAN04530
VAN04540
VAN04550
VAN04560
VAN04570
VAN04580
VAN04590
VAN04600
VAN04610
VAN04620
VAN04630
VAN04640
VAN04650
VAN04660
VAN04670
VAN04680
VAN04690
VAN04700
VAN04710
VAN04720
VAN04730
VAN04740
VAN04750
VAN04760
VAN04770
VAN04780
VAN04790
VAN04800
VAN04810
VAN04820
VAN04830
VAN04840
VAN04850
VAN04860

```

GOTO LOOP;
/* ORDERING OF EQUATIONS */
SUIT:CALL RNG;
/* WE DO SEQ(N+1)=999 TO PRINT IF STATEMENT WITHOUT */
/* EQUATION Y(..)=.. AFTER TEST IF */
SEQ(N+1)=999;
/* PRINTING OF VECTORS 'GIVE' AND 'SEQ' */
PUT SKIP LIST('GIVE','SEQUENCE');
PUT SKIP LIST(' ');
DO I=1 TO N;
PUT SKIP LIST(GIVE(I),SEQ(I));
END;
/* PRINTING OF THE EQUATIONS WITH NEW ORDERING */
DO I=1 TO N+1;
/* FIRST WE LOOK IF THE VARIABLE(SEQ(I)) BELONGS TO THE */
/* 'RECURSIVE PART' OF THE SYSTEM. IF YES(COM(SEQ(I))=1 */
/* IT WILL BE WRITTEN LATER, OTHERWISE(COM(SEQ(I))=0) IT */
/* IS WRITTEN AT ONCE. */
IF (COM(SEQ(I))=0) THEN
CALL IMPRIM(SEQ(I));
END;
/* R1>1 MEANS THAT THERE ARE SOME EQUATIONS IN 'RECURSIVE */
/* PART' OF THE SYSTEM. THESE EQUATIONS ARE IN VECTOR 'REC' */
/* WE PUT THEM IN THE OUTPUT FILE 'TWO' */
IF (R1>1) THEN
DO;
CARD=' ';
CARD='C** END OF SIMULTANEOUS PART ***';
WRITE FILE(TWO) FROM (CARD);
DO I=1 TO R1-1;
CALL IMPRIM(REC(I));
END;
END;
CLOSE FILE(ONE);
CLOSE FILE(TWO);
END VANDERG;

```

VAN04870
VAN04880
VAN04890
VAN04900
VAN04910
VAN04920
VAN04930
VAN04940
VAN04950
VAN04960
VAN04970
VAN04980
VAN04990
VAN05000
VAN05010
VAN05020
VAN05030
VAN05040
VAN05050
VAN05060
VAN05070
VAN05080
VAN05090
VAN05100
VAN05110
VAN05120
VAN05130
VAN05140
VAN05150
VAN05160
VAN05170
VAN05180
VAN05190
VAN05200
VAN05210
VAN05220

REFERENCES

- [1] Bianchi,C., G.Calzolari and P.Corsi, "A Program for Stochastic Simulation of Econometric Models", Econometrica, 46 (1978), 235-236.
- [2] Bianchi,C., G.Calzolari and P.Corsi, "Stochastic Simulation of Econometric Models: Installation Procedures and User's Instructions", IBM Technical Report G513-3568, Pisa, (1978).
- [3] IBM, "Virtual Machine Facility/370: CMS Command and Macro Reference", GC20-1818, IBM, New York, (1976).
- [4] Klein,L.R., "Estimation of Interdependent Systems in Macroeconometrics", Econometrica, 37 (1969), 171-192.
- [5] Van der Giessen,A.A., "Solving Non-Linear Systems by Computer; a New Method", Statistica Neerlandica, 24 (1970), 41-50.